Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Kren

Brezžični vmesnik za Pitotovo cev

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Načrtujte vgrajeni sistem za umerjanje Pitotove cevi na letalih. Vgrajeni sistem naj bo zgrajen na osnovi mikrokrmilnika STM32F407 z jedrom ARM Cortex-M4. Vgrajeni sistem naj s pomočjo brezžičnega vmesnika zajema podatke iz umerjevalne Pitotove cevi in jih prikazuje na zaslonu v pilotski kabini. Za brezžično komunikacijo uporabite radijski modul APLHA-TRX433S.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matevž Kren, z vpisno številko **63080056**, sem avtor diplomskega dela z naslovom:

Brezžični vmesnik za Pitotovo cev

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 26. septembra 2014 Podpis avtorja:

Zahvaljujem se svojemu mentorju, izr. prof. dr. Patriciu Buliću, za pomoč, motivacijo in usmerjanje pri pisanju diplomskega dela. Za pomoč pri razumevanju snovi se zahvaljujem tudi asistentu Roku Češnovarju. Posebna zahvala velja staršem, ki so mi omogočili študij in verjeli vame.

Hvala!

Kazalo

Povzetek

Pitotova cev je priprava za merjenje pritiska, s katero lahko določimo hitrosti tekočin z znano gostoto. Največkrat se uporablja za izračun hitrosti zračnih in vodnih plovil. Pitotove cevi novih modelov letal, ki so spremenjene oblike in jih želimo certificirati, je potrebno umeriti in s tem pokazati, da motnje zračnega pretoka v okolici letala ne vplivajo na natančnost meritev. Cilj diplomskega dela je bila priprava brezžičnega vmesnika, ki bi postopek poenostavil in pohitril. V ta namen je bil razvit vgrajeni sistem za zajem in beleženje podatkov iz senzorjev Pitotove cevi in njihovo prikazovanje v pilotski kabini. Osrednji del brezžičnega vmesnika predstavljata razvojni plošči STM32F4-Discovery, ki prenos podatkov nadzirata z upravljanjem brezžičnih modulov ALPHA-TRX433S.

Ključne besede: Pitotova cev, vgrajeni sistem, STM32F4-Discovery, brezžični vmesnik.

Abstract

A pitot tube is a pressure measurement instrument used to measure fluid flow velocity. Most often, pitot tubes are used to determine the speed of aircraft and boats. Before launching a new aircraft design for use in both private and commercial aviation, it must be shown the correct airspeed data is communicated regardless of airflow disruptions. This is achieved through calibration of the pitot tubes. In this dissertation, we focus on developing a wireless interface to achieve a faster and improved data transmission without fuselage alteration. For this purpose, an embedded system was designed. It integrates the STM32F4-Discovery development environment and the ALPHA-TRX433S high performance radio modules. In essence, data is collected and wirelessly transmitted for display.

Keywords: pitot tube, embedded system, STM32F4-Discovery, wireless interface.

Poglavje 1

Uvod

Letenje je dandanes prepoznano kot daleč najvarnejši in najhitrejši način prevoza. Čeprav se s časom tudi letalska tehnologija močno spreminja, pa osnovne zakonitosti oziroma načini delovanja ostajajo enaki. Za ohranjanje visoke ravni varnosti tako v komercialnem kot tudi v splošnem in vojaškem letalstvu skrbi veliko število sistemov in merilnih naprav. Za merjenje hitrosti se poleg globalnega sistema pozicioniranja (GPS, ang. *Global Positioning System*) uporablja tudi Pitotova oziroma Pitot-Prandtlova cev.

V sodelovanju s podjetjem Pipistrel, ki je vodilni svetovni proizvajalec ultralahkih motorno-jadralnih letal in jadralnih letal s pomožnim motorjem, smo za potrebe umerjanja Pitotovih cevi novih modelov letal pripravili brezžični vmesnik, ki postopek poenostavi in pohitri, obenem pa ne zahteva fizičnega posega v konstrukcijo letala.

Problem smo rešili z uporabo vgrajenega sistema, ki temelji na STM32F4-Discovery razvojni plošči in za prenos podatkov uporablja brezžično komunikacijo. Po uspešnem prenosu se podatki izpišejo na LCD (ang. *Liquid-Crystal Display*) zaslon. Največ dela je bilo vloženega v razumevanje delovanja, programiranje in uporabo brezžičnih modulov.

V prvem delu diplomske naloge bomo predstavili Pitotovo cev, delovanje in možnosti uporabe. Opisali bomo senzor za merjenje razdalj, ki smo ga uporabili z namenom smiselnega preizkušanja vgrajenega sistema. V sklopu

opisa razvojne plošče STM32F4-Discovery bomo omenili tudi procesorsko arhitekturo ARM (ang. *Advanced RISC Machine* oz. *Acorn RISC Machine* pred tem), ki opisuje način delovanja procesorskega jedra Cortex-M4F. Večji del bomo namenili različnim načinom delovanja brezžičnih modulov in pripadajočim nastavitvam. Teoretični del bomo zaključili z opisom LCD zaslona uporabljenega za prikaz podatkov in potrditev uspešnega prenosa podatkov. V drugem delu bomo nadaljevali s predstavitvijo končne rešitve in uporabljenih algoritmov. Osredotočili se bomo tudi na težave, na katere smo tekom dela naleteli. Za zaključek bomo predstavili rezultate dela in možnosti nadaljne uporabe oziroma razširitev.

Poglavje 2

Pitotova cev

Pitotova cev je priprava za merjenje pritiska, s katero lahko določimo hitrosti tekočin z znano gostoto [3]. Tekočina oziroma fluid je skupno ime za podmnožico faz snovi in zajema kapljevine, pline, plazmo in tudi plastične trdnine. Skupna lastnost tekočin je, da se ne upirajo deformaciji. Pitotova cev se najpogosteje uporablja za določitev hitrosti zračnih in vodnih plovil. Imenuje se po francoskemu inženirju Henriju Pitotu, kasnejšo obliko je zasnoval francoski znanstvenik Henry Darcy.

2.1 Načelo delovanja

Osnovna Pitotova cev označuje cev, ki je usmerjena proti smeri toka tekočine. Ker je cev na drugi strani zaprta, se tekočina umiri oziroma se njen tok ustavi. Na tem mestu izmerjeni tlak imenujemo totalni tlak p_t. Ob predpostavki, da je potencialna energija konstantna, lahko z upoštevanjem Bernoullijeve enačbe zapišemo

$$p_t = p_s + \frac{1}{2}\rho v^2. \tag{2.1}$$

Velja, da:

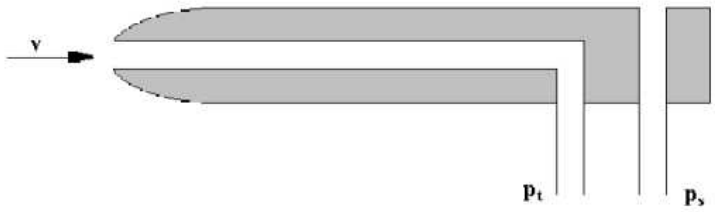
- v je hitrost tekočine z enoto $\frac{m}{s}$,

- p_t je totalni tlak z enoto Pa,
- p_s je statični tlak z enoto Pa,
- ρ je gostota tekočine z enoto $\frac{kg}{m^3}$.

Iz enačbe (2.1) nato izrazimo hitrost tekočine v.

$$v = \sqrt[2]{\frac{2(p_t - p_s)}{\rho}} \tag{2.2}$$

Statični tlak p_s se običajno meri na eni ali obeh straneh plovila. Zaradi varnosti je lahko pomožni sistem merjenja statičnega tlaka, ki sicer ni točen, nameščen tudi v notranjosti plovila. Pitot-Prandtlova cev (glej sliko 2.1) združuje merjenje totalnega in statičnega tlaka z uporabo dodatne cevi, ki leži pravokotno na smer toka tekočine.



Slika 2.1: Pitot-Prandtlova cev

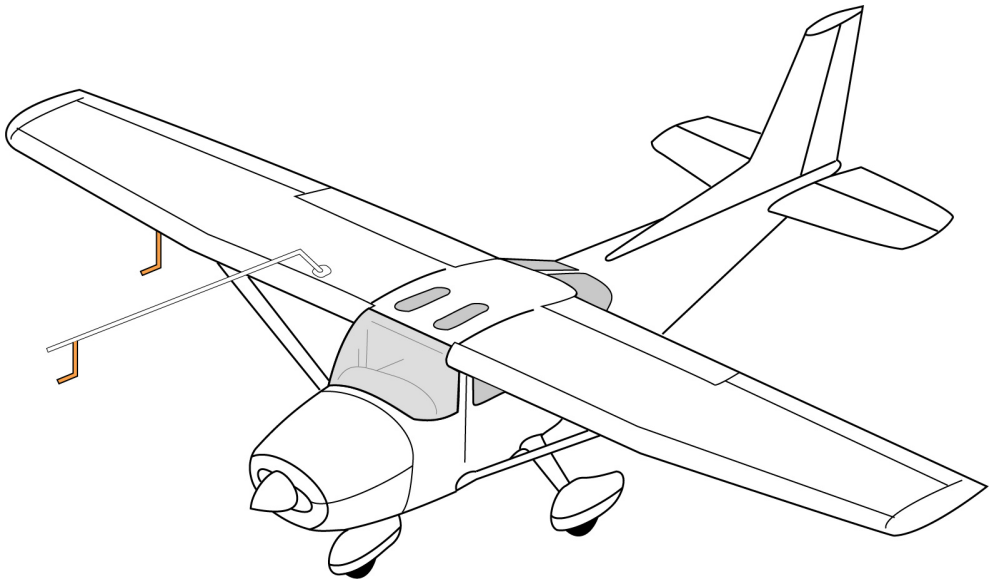
2.2 Merske napake in napake sistema

Napake merjenja se pojavijo zaradi različnih fizikalnih pogojev. Ob spremembah temperature in tlaka v atmosferi se lahko spremeni gostota tekočine, kar negativno vpliva na končni rezultat. Pri visokih hitrostih oziroma višinah se pojavi stisnjenost tekočine, kar je prav tako potrebno upoštevati in temu primerno prilagoditi izračun. Do napačnih meritev lahko pride tudi pri nenadni, zelo hitri spremembi naklona plovila.

Led, insekti ali predmeti lahko Pitotovo cev zamašijo, kar močno ogrozi varnost predvsem v primeru letenja. Čeprav je na primer pri dvigu letala hitrost enakomerna, bo zamašena Pitotova cev napačno označevala povečanje hitrosti. Podobno pri spustu namesto povečanja hitrosti prikaže pojemek hitrosti. V obeh primerih je napačna informacija lahko razlog strmoglavljenja letala. Vsa letala morajo iz teh razlogov imeti sisteme, ki Pitovo cev nadzirajo in jo v primeru nizkih temperatur dodatno ogrevajo [4].

2.3 Umerjanje Pitotove cevi

Pitotove cevi novih modelov letal, ki so spremenjene oblike in jih želimo certificirati, je potrebno umeriti in s tem pokazati, da motnje zračnega pretoka v okolici letala ne vplivajo na natančnost meritev.



Slika 2.2: Skica postopka umerjanja Pitotove cevi

Postopek je izveden s pomočjo dodatne Pitotove cevi (glej sliko 2.2). Ta je nameščena na nastavek, ki sega čez nos letala in izven območja motenj v zračnem pretoku. Dodatna Pitotova cev je namenska in se uporablja le za

meritve tekom preizkušanja posameznega tipa letala. Poleg dveh podatkovnih izhodov, ki podajata vrednosti totalnega in statičnega tlaka, ima cev še dva potenciometra, katerih namen je merjenje kota letala glede na zračni tok. Pridobljeni podatki in nadaljni izračuni služijo v namen umeritve Pitotovih cevi, ki pripadajo letalu.

Poglavje 3

Brezžični vmesnik

Razvit je bil sistem za zajemanje in beleženje podatkov iz senzorjev Pitotove cevi in njihovo prikazovanje v pilotski kabini. Osrednji del brezžičnega vmesnika prestavljata razvojni plošči STM32F4-Discovery, ki z upravljanjem brezžičnih modulov APLHA-TRX433S nadzirata komunikacijo in prenos podatkov.

3.1 Vgrajeni sistemi

Vgrajeni sistem je računalniški sistem namenjen izvajanju specifične funkcije znotraj večjega mehanskega ali električnega sistema. Za razliko od osebnih računalnikov vgrajeni sistemi niso dinamični in največkrat ne omogočajo večje spremembe namembnosti, kar nakazuje njihovo največjo prednost. Odlikujejo se po nižji porabi električne energije in možnosti optimizacije. Poleg glavne procesorske enote lahko vsebujejo mikroprocesorje in procesorje digitalnih signalov.

3.2 Arhitektura ARM

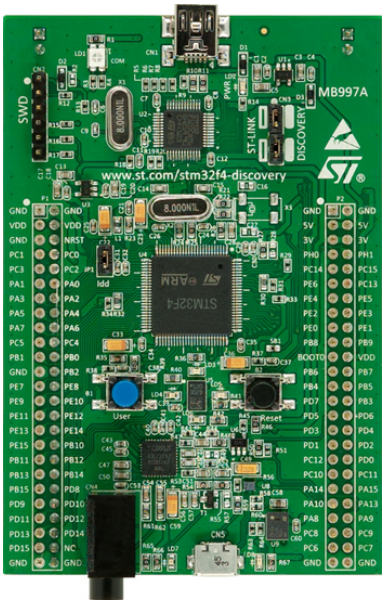
Procesorske arhitekture ARM (ang. *Advanced RISC Machine*, *Acorn RISC Machine* pred tem) predstavljajo teoretične rešitve in načrte izdelave RISC

(ang. *Reduced Instruction Set Computer*) procesorjev podjetja ARM Holdings. Njihova glavna dejavnost je prodaja IP (ang . *Intellectual Property*) jeder, katera se uporabljajo pri izdelavi FPGA (ang. *Field Programmable Gate Array*) in ASIC (ang. *Application-Specific Integrated Circuit*) vezij, tj. mikrokontrolerjev in procesorjev, in se ne ukvarjajo z dejansko proizvodnjo čipov. Najuspešnejša implementacija ARM jedra je ARM7TDMI z več kot nekaj sto milijoni prodanih čipov. Naravnanost k majhnemu številu ukazov procesorjev ARM zahteva uporabo veliko manjšega števila tranzistorjev, kot jih najdemo v običajnih računalnikih. Prednost opisanega pristopa so zmanjšani stroški izdelave, manjše oddajanje toplote in manjša poraba električne energije [1]. Vse to so željene lastnosti pri izdelavi lahkih, prenosnih naprav, npr. pametnih telefonov in tablic. Poenostavljena zgradba in posledična preprostost izvedbe podjetjem tako omogoča izdelavo varčnih SOC (*System On Chip*) komponent za uporabo v vgrajenih sistemih.

3.3 Razvojna plošča STM32F4-Discovery

Razvojni sistem STM32F4-Discovery (glej sliko 3.1) vključuje mikrokrmilnik STM32F407VGT6. Za izračune skrbi visoko zmogljivo procesorsko jedro ARM®Cortex™-M4 z majhnim številom ukazov (RISC, ang. *Reduced Instruction Set Computer*). Procesor ukaze izvaja s hitrostjo do 168 MHz, za naslavljanje pomnilniškega prostora pa se uporablja 32 bitov. Pri računanju uporablja samostojno enoto za računanje s števili predstavljenimi v plavajoči vejici (ang. *Floating Point Unit*).

Za delo je na voljo 1MB delovnega pomnilnika Flash, 192KB delovnega pomnilnika SRAM (ang. *Static Random-Access Memory*) in do 4KB pomožnega pomnilnika SRAM [9]. Mikrokrmilnik poleg dveh krmilnikov DMA (ang. *Direct Memory Access*) omogoča vrsto vhodov, izhodov in vhodno-izhodnih enot. Povezuje jih z dvema vodiloma APB (ang. *Advanced Peripheral Bus*), s tremi vodili AHB (ang. *Advanced High-Performance Bus*) in z 32-bitno matriko, ki omogoča povezavo več vodil AHB. Vse naprave



Slika 3.1: Razvojni sistem STM32F4-Discovery

omogočajo tri 12-bitne analogno-digitalne pretvornike (ADC, ang. *Analog-To-Digital Converter*), dva digitalno-analogna pretvornika (DAC, ang. *Digital-To-Analog Converter*), uro, ki meri trenutni čas (RTC, ang. *Real-Time Clock*), dvanajst splošno namenskih 16-bitnih časovnikov s podporo pulzno-širinske modulacije (PWM, ang. *Pulse-Width Modulation*), dva splošno namenska 32-bitna časovnika in generator naključnih števil (RNG, ang. *Random Number Generator*). Med napravami je na voljo več načinov komunikacije.

- Tri vodila SPI (ang. *Serial Peripheral Interface Bus*)
- Dve vodili CAN (ang. *Controller Area Network*)
- Tri vodila I^2C (ang. *Inter-Integrated Circuit*)
- Dve vodili I^2S (ang. *Integrated Interchip Sound*)
- Dva vmesnika UART (ang. *Universal Asynchronous Receiver/Transmitter*)

- Štirje vmesniki USART (ang. *Universal Synchronous/Asynchronous Receiver/Transmitter*)
- USB OTG (ang. *Universal Serial Bus On-The-Go*)
- Vmesnik SDIO (ang. *Secure Digital Input Output*)

ARM®Cortex™-M4 prištevamo v zadnjo generacijo Armovih procesorkih jeder za uporabo v vgrajenih sistemih. Razvito je bilo v želji po visoko zmogljivem mikrokrmilniku z majhnim številom nožic (ang. *Pin*) in naprednim sistemom za upravljanje s prekinitvami. Ob tem ga dodatno odlikuje nizka poraba električne energije.

3.3.1 Krmilnik vektorskih prekinitev

Krmilnik za delo z gnezdenimi vektorskimi prekinitvami (NVIC, ang. *Nested Vectored Interrupt Controller*) podpira do 240 prekinitev, katerim lahko določimo in dinamično spreminjamo 256 različnih stopenj pomembnosti. Glavno procesorsko jedro in krmilnik NVIC sta izjemno blizu, kar omogoča majhno zakasnitev pri obravnavanju prekinitev in možnost reševanja prekinitev, ki prispejo pozno.

Prekinitve in notranje prekinitve oziroma izjeme so vektorske, zato vsaki prekinitvi pripada točno določen in nespremenljiv prekinitveni vektor [2]. Prekinitveni vektor označuje naslov pomnilniške besede v naslovnem prostoru glavnega pomnilnika in kaže na prvi ukaz prekinitveno servisnega podprograma. Uporabljeni mikrokrmilnik omogoča 82 prekinitvenih vhodov, katere se lahko poljubno omogoči oziroma onemogoči. Vsakemu vhodu lahko določimo eno izmed 16 stopenj prednosti.

3.3.2 Splošno namenske vhodno-izhodne enote

Vsaka splošno namenska vhodno-izhodna enota (GPIO, ang. *General Purpose Input/Output*) ima štiri 32-bitne nastavitvene registre (GPIOx_MODER,

GPIOx_OTYPER, GPIOx_OSPEEDR in GPIOx_PUPDR), dva 32-bitna podatkovna registra (GPIOx_IDR in GPIOx_ODR), 32-bitni register za nastavljanje oziroma brisanje vrednosti (GPIOx_BSRR), 32-bitni register za zaklepanje (GPIOx_LCKR) in dva 32-bitna registra (GPIOx_AFRH and GPIOx_AFRL) AF (ang. *Alternative Function*) [8]. Ima 16 vhodnih oziroma izhodnih nožic, s katerimi lahko upravljamo in jim nastavljamo lastnosti, kot so hitrost, upor za dvig ali spust nivoja, stanje in druge.

3.3.3 Možnosti uporabe

Družina mikrokrmilnikov STM32F407VG označuje vrsto naprav oziroma razvojnih sistemov z različnimi števili nožic in vhodno-izhodnimi enotami, ki so uporabnikom na voljo. Iz tega razloga lahko mikrokrmilnike uporabljamo v različne namene.

- Alarmni sistemi
- Zvočni sistemi
- Digitalni računalniki, namenjeni avtomatizaciji elektromehaničnih procesov, npr. strojev v industriji (PLC, ang. *Programmable Logic Controller*)
- Nadzor aplikacij
- Upravljanje s stroji v medicini
- Optični čitalci in tiskalniki
- Brezžični vmesniki
- Nadzor pogonskih motorjev

3.4 Modul za brezžično komunikacijo ALPHA-TRX433S

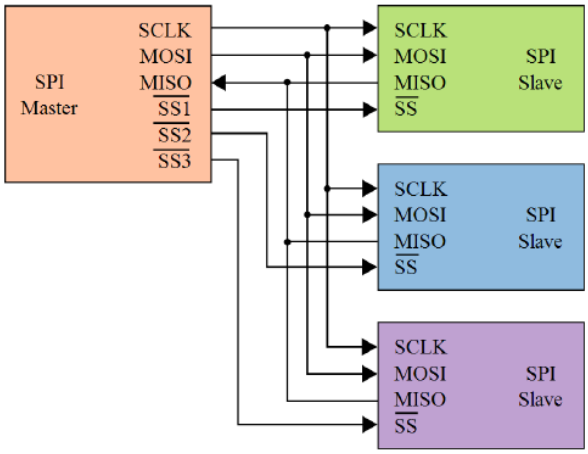
Brezžično pošiljanje in sprejemanje podatkov je bilo uresničeno z moduloma za brezžično komunikacijo ALPHA-TRX433S (glej sliko 3.2). Brezžični modul odlikujeta predvsem visoka zmogljivost in izjemno nizka poraba električne energije. V stanju mirovanja se modul ob sprejemu podatkov samodejno prebudi in sproži komunikacijo z mikrokrmilnikom. Možnosti uporabe vključujejo brezžične varnostne sisteme, signale, ki opozarjajo na napake, brezžične vmesnike nadzora široke palete naprav, npr. garažnih vrat, in branje in sporočanje senzorskih vrednosti.

Podatkovni prenos deluje na način frekvenčne modulacije signala FSK (ang. *Frequency-Shift Keying*), pri čemer odstopanje od sredinske frekvence predstavlja logično vrednost bita [7]. Prenos lahko doseže hitrosti do 115Kb na sekundo in se uspešno zaključi na razdaljah tudi do 300m. ALPHA-TRX433S lahko oddaja in sprejema na različnih frekvenčnih vrednostih. Komunikacijo smo izvedli na frekvencah 433MHz, 868MHz in 915MHz.

Slika 3.2: Modul za brezžično komunikacijo ALPHA-TRX433S

3.4.1 Vmesnik SPI

Komunikacija z razvojnim sistemom STM32F4-Discovery, ki podatke ali sprejema ali pošilja, poteka preko vodila SPI (ang. *Serial Peripheral Interface*). Vmesnik SPI opisuje sinhrono serijsko podatkovno povezavo elektronskih naprav, ki deluje v dvosmernem načinu. Protokol določa napravo, ki je nadre-



Slika 3.3: Signalne linije protokola SPI

jena in komunikacijo proži, in napravo oziroma več naprav, ki so podrejene. Pravimo, da protokol deluje na način gospodar/suženj (ang. *Master/Slave*).

Uporablja se štiri signalne linije (glej sliko 3.3).

- Nadrejena naprava posluša, podrejena naprava sporoča (MISO, ang. *Master Input Slave Output*).
- Nadrejena naprava sporoča, podrejena naprava posluša (MOSI, ang. *Master Output Slave Input*).
- Urin takt, ki ga oddaja gospodar (CLK ali SCK, ang. *Clock*).
- Izbira naprave, s katero želi gospodar vzpostaviti povezavo (SS, ang. *Slave Select*). Signal je aktiven v nizkem stanju.

3.4.2 Sprejemanje in pošiljanje podatkov ter nastavitve

Z modulom ALPHA-TRX433S upravljamo izključno s pisanjem in istočasnim branjem 16-bitnih vrednosti. Vsi ukazi so iste dolžine, pri čemer najpomembnejših osem bitov (bit 15 - bit 8) največkrat enolično določa izbiro ukaza,

ostalih osem bitov pa informacijo oziroma nastavitev, ki jo želimo z ukazom uresničiti.

Frekvenčno območje in način delovanja

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	0	0	el	ef	b1	b0	x3	x2	x1	x0

S spreminjanjem logičnih vrednosti bitov $b1$ in $b0$ izbiramo frekvenčno območje. Nastavljeni bit *el* omogoči register za pošiljanje (TX), nastavljeni bit ef pa omogoči delovanje vmesnega pomnilnika (ang. *Buffer*) vrste FIFO (ang. *First In First Out*) v načinu delovanja sprejemanja podatkov.

Poraba električne energije

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	1	0	er	ebb	et	es	ex	eb	ew	dc

Z nastavitvijo bita *er* vklopimo način sprejemanja podatkov. Nastavljeni bit *et* pomeni način oddajanja podatkov. Bit *ew* skrbi za vklop časovnika, ki brezžični modul samodejno prebudi iz stanja mirovanja.

Frekvenca delovanja

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	0	f11	f10	f9	f8	f7	f6	f5	f4	f3	f2	f1	f0

Nastavljanje logičnih vrednosti bitov $f11$-$f0$ nam omogoča večjo natančnost frekvence delovanja modula. Če je izbrana začetna frekvenca 433MHz, lahko vrednost spreminjamo koračno po 2.5KHz. V primeru izbrane začetne frekvence 868MHz je možnost najmanjše spremembe 5KHz, pri začetni frekvenci 915MHz pa 7.5KHz.

Hitrost prenosa podatkov

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	9	1	1	9	cs	r6	r5	r4	r3	r2	r1	r0

Spreminjanje logičnih vrednosti bitov $r6$-$r0$ nam omogoča izbiro hitrosti prenosa podatkov (ang. *Bit Rate*).

$$BitRate = \frac{10^7}{29} * \frac{1 + cs * 7}{R + 1} \tag{3.1}$$

Nastavitve sprejemnika

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	0	p16	d1	d0	i2	i1	i0	g1	g0	r2	r1	r0

Z brisanjem logične vrednosti bita $p16$ spreminjamo namembnost 16. nožice modula, ki namesto izhoda VDI (ang. *Valid Data Indicator*) postane prekinitveni vhod modula. Nastavljanje bitov $d1$-$d0$ vpliva na odzivni čas signalne linije VDI. S spreminjanjem bitov $i2$-$i0$ določamo razpon odstopanja od sredinske frekvence, pri čemer še sprejemamo podatke. Nastavitve bitov $g1$-$g0$ predstavljajo možnost ojačitve signala (LNA, ang. *Low Noise Gain*) v okolju z motnjami.

Nizkoprepustni oziroma visokoprepustni filter

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	0	1	0	al	ml	1	s	1	f2	f1	f0

S spreminjanjem logične vrednosti bita s izbiramo tip filtra, ki ga želimo uporabiti. Logična ničla predstavlja digitalni filter, logična enica pa analogni filter RC (ang. *Resistor–Capacitor*). Frekvenčna prepustnost se nastavlja z

zunanjo vezavo kondenzatorja. Z bitoma $f1$ in $f0$ določamo mejne vrednosti prepoznavanja kvalitete podatkov (DQD, ang. *Data Quality Detector*).

Vrsta FIFO in ponastavitev modula

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	1	0	1	0	f3	f2	f1	f0	0	al	ff	dr

Z nastavljanjem bitov $f3$-$f0$ določimo število zahtevanih podatkovih bitov, pri čemer se proži prekinitev in zahteva po branju podatkov. Logična enica bita *al* pomeni, da se za izpolnjen pogoj proženja prekinitve upošteva tudi bite, ki pripadajo postopku usklajevanja oziroma sinhronizacije tekom komunikacije. Vrednost bita *sp* določa dolžino postopka usklajevanja, pri čemer logična enica pomeni, da je za sinhronizacijo potreben je 1B podatkov. Določitev bita ff pomeni, da bo vrsta FIFO omogočena šele po uspešno opravljeni sinhronizaciji. Nastavljeni bit *dr* onemogoči ponastavitveni način delovanja modula v primeru nepričakovane spremembe napetosti (600mV).

Podatkovni vzorec postopka sinhronizacije

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	1	1	1	0	b7	b6	b5	b4	b3	b2	b1	b0

Nastavljanje oziroma brisanje bitov $b7$-$b0$ nam omogoča spreminjanje podatkovnega vzorca bitov uporabljenega tekom postopka sinhronizacije.

Branje podatkov iz vrste FIFO

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Ukaz pomeni branje podatkov, ki po proženju prekinitve čakajo v vrsti FIFO.

Samodejna izbira frekvence

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	0	0	a1	a0	rl1	rl0	st	fi	oe	en

Z ukazom izbiro frekvence prepustimo sistemu. Z bitoma $rl1$ in $rl0$ nastavimo dovoljene mejne vrednosti.

Nastavitve oddajnika

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	0	0	mp	m3	m2	m1	m0	0	p2	p1	p0

Z nastavljanjem oziroma brisanjem bita mp določimo polarnost frekvenčne modulacije signala. S spreminjanjem logičnih vrednosti bitov $m3$-$m0$ nastavimo frekvenčno odstopanje od sredinske frekvence. To lahko zajema vrednosti od 15KHz do 240KHz. Določanje bitov $p2$-$p0$ vpliva na oddajno moč pošiljanja podatkov z enoto dBm.

Pošiljanje podatkov

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	1	0	0	0	t7	t6	t5	t4	t3	t2	t1	t0

Z ukazom v register za pošiljanje zapišemo 1B podatkov in jih brezžično prenesemo.

Časovnik in stanje mirovanja

Z nastavljanjem bitov $m7$-$m0$ in bitov $r4$-$r0$ določimo časovno okno v ms, po preteku katerega časovnik modul prebudi iz stanja mirovanja. Ob koncu vsakega obhoda časovnega okna je potrebno nastaviti in brisati logično vrednost bita et.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	r4	r3	r2	r1	r0	m7	m6	m5	m4	m3	m2	m1	m0

$$T = M * 2^R \tag{3.2}$$

Zaznavanje nizkega nivoja električne energije baterije in delilnik izhodne ure

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	0	0	0	d2	d1	d0	0	v3	v2	v1	v0

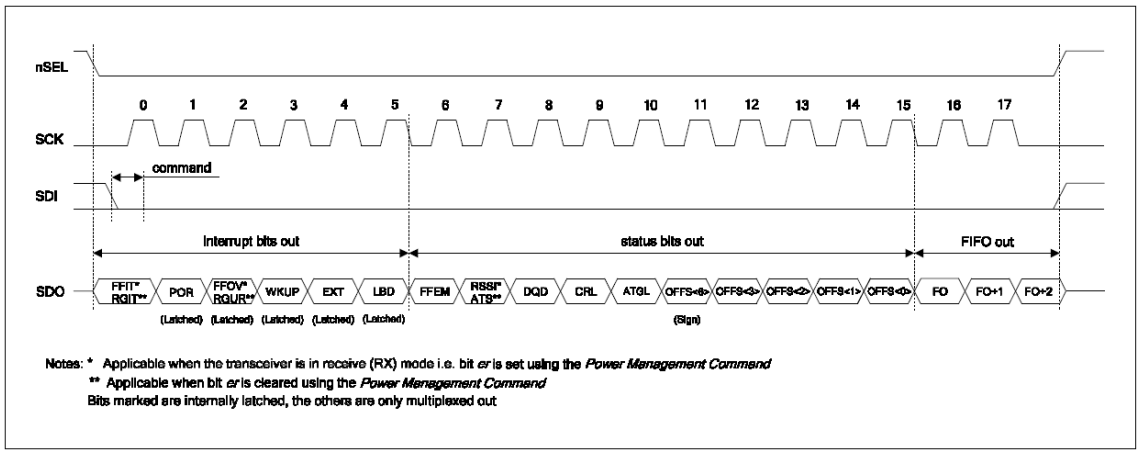
S spreminjanjem logičnih vrednosti bitov $v4$-$v0$ določimo mejhno vrednost napetosti, ob kateri se proži opozorilo o nizkem nivoju električne energije uporabljene baterije. Določanje bitov $d2$-$d0$ pomeni frekvenco izhodne ure z enoto MHz. Ta obsega vrednosti od 1MHz do 10MHz.

Branje statusnega registra

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Ukaz za branje in istočasno brisanje večjega dela statusnega registra se kot edini začne z logično vrednostjo 0. Modul odgovor vrne preko SDO (ang. *SPI Data Output*) oziroma MISO signalne linije (glej sliko 3.4). Postavitev bitov 15-10 istočasno proži izhodni prekinitveni signal, ki ga lahko zaznamo na nIRQ nožici modula in ga označuje logična ničla.

Nastavljeni bit 15 pomeni, da je oddajnik pripravljen na pošiljanje naslednjega zaporedja podatkov oziroma, če modul deluje v načinu sprejemanja, da so podatki v vrsti FIFO pripravljeni na branje. Logična enica bita 14 označuje stanje modula po ponastavitvi sistema. Če je nastavljen bit 13,



Slika 3.4: Branje statusnega registra

pomeni, da je pošiljanje podatkov hitrejše od pisanja podatkov v register za pošiljanje (ang. *Register Under Run*). V načinu sprejemanja podatkov, nastavljeni bit 13 pomeni, da je bilo sprejetih več podatkov, kot jih podpira vrsta FIFO. Logična enica bita 12 označuje prekinitev časovnika, ki skrbi za bujenje modula iz stanja mirovanja. Nastavljeni bit 11 označuje vhodno prekinitveno zahtevo, ki jo modulu pošlje mikrokrmilnik. Logična enica bita 10 pomeni nizek nivo električne energije uporabljene baterije. Bit 9 označuje izpraznjeno vrsto FIFO, bit 6 pa sprejemanje podatkov, ki ustrezajo nastavljenim zahtevam kvalitete.

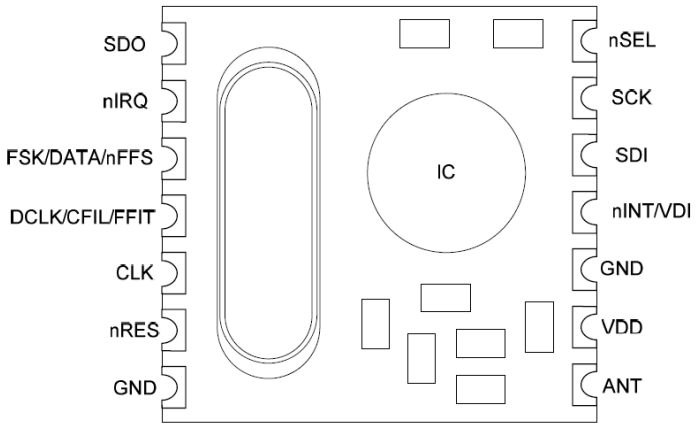
3.4.3 Povezave nožic modula

Modul vključuje 14 nožic (glej sliko 3.5). Večino teh je potrebno upravljati, nekaj pa jih glede na izbran način delovanja modula lahko zanemarimo.

- VDD (zahtevano) - Napajalna napetost od 2.2V do 3.8V.
- GND (zahtevano) - Ozemljitev
- nRES (zahtevano) - Vhodni signal, ki je aktiven ob nastavitvi logične ničle. Če želimo, da modul deluje in se ne ponastavlja, mora biti vhod

nastavljen na logično enico.

- nINT/VDI (poljubno) - Nožica lahko deluje kot izhod, ki opozarja na ustreznost sprejetih podatkov. Kot vhod označuje vhodno prekinitveno zahtevo.
- nIRQ (poljubno) - Izhodni prekinitveni signal
- SDI (zahtevano) - Vhodna podatkovna linija oziroma signal MOSI
- SDO (zahtevano) - Izhodna podatkovna linija oziroma signal MISO
- SCK (zahtevano) - Urin takt, ki ga oddaja mikrokrmilnik in skrbi za komunikacijo SPI.
- nSEL (zahtevano) - Signal protokola SPI za izbiro naprave



Slika 3.5: Prikaz nožic modula

- FSK/DATA/nFFS (poljubno, sicer zahteva 10KΩ upor za spust nivoja) - Odvisno od načina uporabe modula lahko predstavlja vhodni signal podatkov, ki jih želimo poslati, izhodni signal podatkov, ki smo jih sprejeli, ali izbiro vrste FIFO.

- DCLK/CFIL/FFIT (poljubno) - Odvisno od načina uporabe modula lahko predstavlja podatkovno uro, ko vrsta FIFO ni uporabljena, povezavo za zunanji kondenzator, ali označuje, da so podatki v vrsti FIFO pripravljeni na branje.
- CLK (poljubno) - Izhodni urin takt
- ANT (zahtevano) - Povezava, na katero vežemo anteno.

Načini uporabe

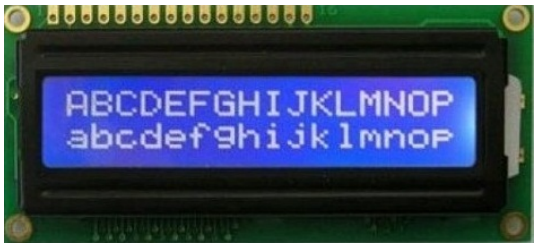
Za nadzor in brezžični prenos podatkov so vedno uporabljene le signalne linije nSEL, SCK, SDI in SDO, pri čemer se SDO nožica uporablja tudi kot izhodni prekinitveni signal, ki mikrokrmilnik v načinu sprejemanja podatkov opozori na prekinitvene zahteve v povezavi vrste FIFO.

Če želimo prenos pohitriti, lahko uporabimo dodatne vhodne oziroma izhodne nožice. Najhitrejši način prenosa podatkov vključuje izhod FFIT in vhodno linijo nFFS ter omogoča neposredni dostop do podatkov v vrsti FIFO.

3.5 Zaslon LCD QY-1602A in senzor za določitev oddaljenosti GP2Y0

Namesto Pitove cevi in Pipistrelovega instrumenta P500, ki sta sicer del brezžičnega vmesnika, smo za potrebe diplomskega dela uporabili senzor razdalj GP2Y0, prikaz podatkov pa smo s pomočjo že napisane programske knjižnice izvedli na zaslonu LCD QY-1602A (glej sliko 3.6).

Zaslon prikazuje 16 znakov v vsaki izmed dveh vrstic. Na voljo sta dva načina uporabe, pri čemer je razlika izključno v hitrosti izpisa. Odločili smo se za način delovanja, ki uporablja štiri izmed osmih podatkovnih signalnih linij, počasnejši pa je zaradi potrebe dveh dostopnih ciklov tekom enega izpisa [5]. Nudi množico znakov ASCII 7 na naslovih od 0x20 do 0x7f, tej pa sledi razširjen nabor znakov na naslovih od 0x80 do 0xff (glej sliko 3.7).



Slika 3.6: Zaslon QY-1602A

Slika 3.7: Znaki, ki jih zaslon lahko prikaže.

Poleg štirih podatkovnih nožic (11-14) zaslon za delovanje potrebuje 5V napajanje in ozemljitev. Enako napajanje namenimo osvetlitvi naprave in 10KΩ potenciometru, ki je vezan na tretjo nožico naprave in skrbi za spreminjanje kontrasta. Nožica 4 določa izbiro registra (ang. *Register Select*). Uporablja se za izbiro podatkovnega ali ukaznega načina delovanja. Nožica 5, katere uporaba je poljubna, nakazuje, da je zaslon bodisi zaseden bodisi pripravljen na nov prenos podatkov. Če je ne uporabljamo, je zahtevana ozemljitev nožice. Nožica 6 označuje signalno linijo, ki sprovede branje oziroma pisanje podatkov.

Senzor za določitev oddaljenosti (PSD, ang. *Position Sensitive Device*) GP2Y0 za delovanje uporablja infrardeče valovanje oziroma svetlobo in je natančen na razdaljah od 20cm do 120cm [10]. Zahteva napajanje 5V, ozemljitev in vključuje izhodno nožico. Podatek o izmerjeni razdalji v mersko enoto pretvorimo s pomočjo analogno-digitalnega pretvornika.

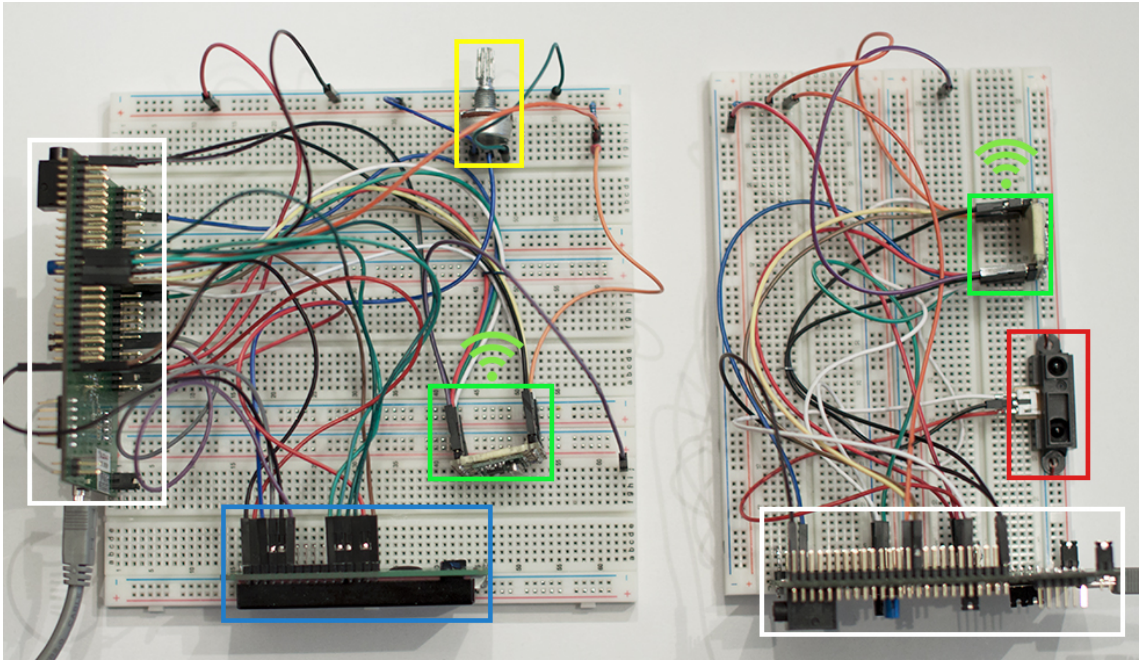
Poglavje 4

Izvedba in programska koda

Programsko kodo smo pripravili v razvijalskem okolju *IAR Embedded Workbench*. Za potrebe diplomskega dela je bila ustrezna zastonjska licenca, ki je sicer omejena z velikostjo projekta. Razvijalsko okolje je vodilno v svetu C/C++ prevajalnikov in razhroščevalnih orodij pri delu z 8-, 16- in 32-bitnimi procesorji in nudi podporo številnim proizvajalcem razvojnih sistemov oziroma mikrokrmilniških računalnikov.

Brezžični vmesnik predstavljata dve nepovezani plošči (glej sliko 4.1). Desni del sistema označuje oddajnik, kjer je z belo barvo označena razvojna plošča STM32F4-Discovery. Z rdečo barvo je označen senzor za določitev oddaljenosti, z zeleno barvo pa modul ALPHA-TRX433S, ki skrbi za brezžično komunikacijo.

Plošča na levi predstavlja sprejemnik sistema. Z belo barvo je označena razvojna plošča, ki upravlja z brezžičnim modulom in prejete podatke izpiše na zaslon. Z modro barvo je označen zaslon LCD, z rumeno barvo pa je označen potenciometer, s katerim nadziramo kontrast prikaza na zaslonu.



Slika 4.1: Shema sistema

4.1 Uporaba senzorja za določitev oddaljenosti

V namen izpopolnjenosti sistema smo za postopek branja in pretvorbe analogne vrednosti senzorja uporabili krmilnik DMA (ang. *Direct Memory Access*), enega izmed načinov optimizacije vhodno-izhodnega dela računalnika. Krmilnik poskrbi za delo z vhodno-izhodnimi napravami in jim omogoči dostop do glavnega pomnilnika brez obremenjevanja glavne procesorske enote, ki lahko v tem času opravlja druga dela. Izboljšava bi bila potrebna v primeru, če bi uporabljali več naprav.

Po inicializaciji in začetni nastavitvi krmilnika DMA z ukazom *ADC_SoftwareStartConv(ADC3)* pričnemo analogno-digitalno pretvorbo na izhodu senzorja izmerjene napetosti. Uporabljena programska koda, ki sicer ni vključena, je bila uresničena s pomočjo primera vključenega v razvojno okolje. Vrednosti se shranjujejo na pomnilniški naslov *ADC3_DR_ADDRESS*,

ki je naslovljen v funkciji *ADC3_CH12_DMA_Config()*.

```
#define ADC3_DR_ADDRESS     ((uint32_t)0x4001224C)
#define treshold 70
#define size 10

void ADC3_CH12_DMA_Config();
__IO uint16_t ADC3ConvertedValue[1];

int main(void){
  ADC3_CH12_DMA_Config();
  ADC_SoftwareStartConv(ADC3);
}
```

Razvit je bil algoritem, ki je izločeval ektremne vrednosti pretvorb, vendar zaradi zelo enakomernega delovanja senzorja ni potreben. V neskončni zanki nato preverjamo ali novo izmerjena vrednost od prejšnje odstopa za več, kot je določena vrednost oznake *treshold*. Rezultat pred pošiljanjem paketa pretvorimo v centimetre.

```
  while(1){
    now = ADC3ConvertedValue[0];
    if (now > last){
      if (now - last > treshold){
        last = now;
        //poslji
        now = 0 - (1.5 + 0.01715 * now * 0.9) + 38;
        if (now <= 1 || now > 255){
          now = 1;
        }
        send_packet(now);
      }
    }
    else{
      if (last - now > treshold){
        last = now;
        //poslji
        now = 0 - (1.5 + 0.01715 * now * 0.9) + 38;
        if (now <= 1 || now > 255){
```

```
20            now = 1;
21          }
22          send_packet(now);
23        }
24      }
25      delay(200000);
26    }
```

4.2 Brezžična komunikacija in nadzor modulov ALPHA-TRX433S

Največ dela je bilo vloženega v programski del vzpostavitve uspešne brezžične komunikacije. Eden glavnih razlogov je površno napisana dokumentacija proizvajalca uporabljenih modulov, pri čemer je velika večina primerov, ki so že na voljo, napisanih v zbirnem jeziku.

Brezžično komunikacijo začnemo z inicializacijo nožic in časovnika razvojnega sistema. Nadaljujemo z nastavitvami, ki so potrebne za krmiljenje modulov po protokolu SPI. Po krajši zakasnitvi brišemo statusni register modula in izvedemo potrebno inicializacijo, pri čemer določimo, ali modul podatke oddaja ali sprejema.

```
1 #include "stm32f4xx.h"
2 #define treshold 70
3 #define size 10
4
5   SPI_and_GPIO_Init();
6   init_TIM4_OC();
7   delay(100000); //100 ms
8   send_CMD(0x0000);
9   init_module(1);
```

Sledi primer inicializacije in nastavitve ene izmed nožic uporabljene v namen komunikacije SPI.

```
1 void SPI_and_GPIO_Init(){
```

```
  GPIO_InitTypeDef GPIO_InitStructure;
  /* Omogoci SCK, MOSI in MISO GPIO uro */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA , ENABLE);

  /* Nastavi nozico GPIO za izbiro naprave, ki poslusa */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  GPIO_SetBits(GPIOA, GPIO_Pin_8);
}
```

Urin takt časovnika je 84MHz. Če želimo meriti čas ene mikrosekunde, moramo urin takt najprej deliti, nato pa nastaviti register ARR. Časovnik nato s spremenjeno frekvenco šteje do vrednosti registra ARR in štetje ponavlja. Ko je v registru CRR zapisana vrednost dosežena, se nastavi zastavica uporabljenega kanala in nas opozori na izbrano časovno zakasnitev. V predstavljenem primeru urin takt delimo s številom 1, vrednost registra ARR pa je 84. V času ene sekunde se zastavica torej postavi milijonkrat.

```
void init_TIM4_OC(){
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_OCInitTypeDef TIM_OCInitStructure;
  //vklopimo uro APB1 za TIM4
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

  TIM_TimeBaseStructure.TIM_Period = 84; // TIM4_ARR
  TIM_TimeBaseStructure.TIM_Prescaler = 0; //delitelj
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV2;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);

  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_Toggle;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;

```

```
17   //nastavimo vrednost registra CCR v 'prvem' kanalu
18   TIM_OCInitStructure.TIM_Pulse = 84;
19   TIM_OC1Init(TIM4, &TIM_OCInitStructure);
20
21   TIM_Cmd(TIM4,ENABLE);
22 }
```

Zakasnitev je uresničena z uporabo opisanega časovnika. Po branju zastavice je le-to potrebno ponastaviti.

```
1 void delay(int microseconds){
2   int counter = 0;
3   while (counter < microseconds){
4     if (TIM_GetFlagStatus(TIM4,TIM_FLAG_CC1)){
5       counter++;
6       TIM_ClearFlag(TIM4,TIM_FLAG_CC1);
7     }
8   }
9 }
```

V nadaljevanju je predstavljena programska knjižnica za upravljanje in nadzor brezžičnih modulov. Vsi ukazi mikrokrmilnika se pošiljajo preko vmesnika SPI. Zaradi načina delovanja modulov in uporabe le štirih signalnih linij smo se odločili za programsko realizacijo komunikacije SPI.

Komunikacijo SPI začnemo s pošiljanjem najpomembnejšega izmed 16 bitov ukaza. Obenem beremo vhodno podatkovno linijo MISO in v spremenljivko *temp* shranjujemo odgovor izbrane naprave, tj. modula. Komunikacijo zaključimo z nastavljenim izhodom nSEL, ki označuje konec naslavljanja naprave.

```
1 uint16_t send_CMD(uint16_t cmd){
2   int i;
3   uint16_t temp=0;
4   GPIO_ResetBits(GPIOB, GPIO_Pin_7); //SCK->0
5   GPIO_ResetBits(GPIOA, GPIO_Pin_8); //nSEL->0
6   for(i=0;i<16;i++){ //vsi ukazi so 16-bitni
7     if(cmd&0x8000){
8       GPIO_SetBits(GPIOB, GPIO_Pin_5); //MOSI -> 1
```

```
9       }
10      else{
11        GPIO_ResetBits(GPIOB, GPIO_Pin_5); //MOSI -> 0
12      }
13      delay(5);
14      GPIO_SetBits(GPIOB, GPIO_Pin_7); //SCK -> 1
15      delay(5);
16      temp<<=1;
17      if(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_4)){ //MISO
18        temp|=0x0001;
19      }
20      GPIO_ResetBits(GPIOB, GPIO_Pin_7); //SCK -> 0
21      cmd<<=1;
22    }
23    GPIO_SetBits(GPIOA, GPIO_Pin_8); //nSEL -> 1
24    return(temp);
25  }
```

Inicializacija označuje zaporedje poslanih ukazov, s katerimi nastavimo delovanje modulov.

```
1   void init_module(int type){
2     GPIO_SetBits(GPIOA, GPIO_Pin_8); //nSEL->1
3     GPIO_ResetBits(GPIOB, GPIO_Pin_7); //SCK->0
4     GPIO_ResetBits(GPIOB, GPIO_Pin_5); //MOSI->0
5     delay(300);
6
7      if (type == 1){
8        send_CMD(0x80A7); // oddajnik
9      }
10     else{
11       send_CMD(0x8067); // sprejemnik
12     }
13
14     // General Init
15     send_CMD(0x8208); // poraba elektricne energije
16     send_CMD(0xA680); // frekvenca delovanja
17     send_CMD(0xC647); // hitrost prenosa podatkov
```

```
18    send_CMD(0x94A0); // nastavitev sprejemnika
19    send_CMD(0xC2AB); // nizkoprepustni oziroma visokoprepustni
          filter
20    send_CMD(0xCA81); // vrsta FIFO in ponastavitev modula
21    send_CMD(0xCED4); // podatkovni vzorec postopka
          sinhronizacije
22    send_CMD(0xC4F7); // samodejna izbira frekvence
23    send_CMD(0x9850); // nastavitev oddajnika
24    send_CMD(0xE000); // casovnik in stanje mirovanja
25    send_CMD(0xC800); // nacin delovanja, ki porabi manj
          elektricne energije
26    send_CMD(0xC040); /* zaznavanje nizkega nivoja elektricne
27                         energije baterije in delilnik izhodne ure */
28 }
```

Funkcija *wait_for_interrupt()* bere podatkovni vhod MISO. Logična enica označuje prekinitveno zahtevo in s tem možnost ponovnega pošiljanja podatkov oziroma branje vrste FIFO, če je modul v načinu delovanja pošiljanja.

```
1 void wait_for_interrupt(){
2   GPIO_ResetBits(GPIOA, GPIO_Pin_8); // nSEL->0
3   while(!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_4)){
4   }
5 }
```

Tako pošiljanje kot tudi branje podatkov uresničimo z uporabo enega izmed ukazov predstavljenih v tretjem poglavju.

```
1 uint16_t read(void){
2    uint16_t data;
3    wait_for_interrupt();
4    data = send_CMD(0xB000);
5    return(data&0x00FF);
6 }
```

```
1 void write(uint8_t abyte){
2    wait_for_interrupt();
3    send_CMD(0xB800 + abyte);
4 }
```

Uspešen brezžični prenos oziroma pošiljanje podatkov se začne z nastavitvijo oddajnika in sinhronizacijo, ki je odvisna od nastavitev obeh modulov. Sprejemnik sinhronizacijske bajte prepozna in jih zavrže. Ob koncu prenosa podatkov še enkrat pošljemo prazne bajte in s tem zagotovimo, da se je prenos vseh podatkov resnično končal.

```
void send_packet(uint8_t data){
    uint16_t i;
    send_CMD(0x8238); //nastavitev oddajnika
    write(0xAA);
    write(0xAA);
    write(0xAA);
    write(0x2D);
    write(0xD4);
    for (i=0;i<size;i++){
        write(data);
    }
    write(0xAA);
    write(0xAA);
    write(0xAA);
    wait_for_interrupt();
    send_CMD(0x8208);
}
```

Sprejem podatkov prav tako poteka v obliki zaporedja ukazov, ki jih mikrokrmilnik pošlje brezžičnemu modulu. Klic funkcije *read()* v zanki čaka na prekinitev, ki označuje uspešen sprejem podatkov in zahtevo po branju vrste FIFO.

```
void receive_packet(){
    uint16_t i;
    send_CMD(0x82C8); //nastavitev sprejemnika
    send_CMD(0xCA81); //omogoci vrsto FIFO
    send_CMD(0xCA83); //izprazni vrsto FIFO
    for (i=0;i<size;i++)    {
        tab[i]=read();
    }
    send_CMD(0x8208);
```

```
10 }
```

4.3 Prikaz podatkov na zaslonu

Za izpis podatkov na zaslon smo uporabili že pripravljeno knjižnico, ki združuje razvojni sistem STM32F4-Discovery in standardni zaslon 2x16 [6]. Po prejetju podatkov z enostavnim izračunom ASCII vrednosti število pretvorimo v znakovni zapis in ga posredujemo zaslonu.

```
1 #include "stm32f4xx.h"
2 #include "stm32_ub_lcd_2x16.h"
3 #define size 10
4
5 int main(void){
6   char smth[13];
7   uint8_t tab[size];
8   UB_LCD_2x16_Init();
9   delay(500000);
10  init_module(0);
11  while(1){
12    receive_packet();
13    int temp = tab[0];
14    smth[3] = (char)(temp/10 + 48);
15    smth[4] = (char)(temp%10 + 48);
16    UB_LCD_2x16_String(1,0,"Curr. distance");
17    UB_LCD_2x16_String(3,1, smth);
18  }
19 }
```

Poglavje 5

Zaključek

V diplomskem delu smo razvili vgrajeni sistem za brezžično komunikacijo in beleženje oziroma izpisovanje podatkov. Brezžični vmesnik temelji na razvojnem sistemu STM32F4-Discovery in brezžičnem modulu ALPHA-TRX433S. Sistem odlikuje visoka zmogljivost, zanesljivost brezžičnega prenosa podatkov in nizka poraba električne energije. Čeprav je bil končni cilj pohitritev in poenostavitev postopka umerjanja Pitotovih cevi na novih modelih letal, so možnosti uporabe in nadaljnega razvoja veliko obsežnejše. Zaradi uporabe krmilnika DMA je upravljanje vhodno-izhodnih naprav enostavno in hitro. Iz tega razloga bi izdelano rešitev lahko uporabili za vrsto merilnih naprav, ki vključujejo številne senzorje in bi zaradi enostavne komunikacije pridobile na vrednosti.

Tekom izdelave diplomske naloge smo spoznali načelo delovanja Pitotove cevi in uporabe le-te v praksi. Podrobno smo preučili in opisali razvojno ploščo STM32F4-Discovery in brezžični modul ALPHA-TRX433S, med katerima komunikacija poteka s pomočjo vmesnika SPI. Izvedli smo preizkuse uspešnosti brezžičnega prenosa podatkov in napisali programsko kodo za uporabo in krmiljenje brezžičnih modulov. Prav tako smo v namen prikaza delovanja uporabili senzor za določanje razdalje, ki oddaljenost meri s pomočjo infrardečega valovanja.

Literatura

[1] W. Hohl. "ARM Assembly Language: Fundamentals and Techniques", CRC Press, 2009.

[2] J. Yiu. "The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors, Third Edition", Newnes, 2013.

[3] (2014) Opis Pitotove cevi na spletnem mestu Wikipedia. Dostopno na http://en.wikipedia.org/wiki/Pitot_tube in http://sl.wikipedia.org/wiki/Pitot-Prandtlova_cev.

[4] (2014) Opis merskih napak Pitotove cevi na spletnem mestu Wikipedia. Dostopno na http://en.wikipedia.org/wiki/Pitot-static_system.

[5] (2014) Opis delovanja zaslonov LCD. Dostopno na http://www.utasker.com/docs/uTasker/uTaskerLCD.PDF.

[6] (2014) Programska knjižnica za uporabo zaslona LCD. Dostopno na http://mikrocontroller.bplaced.net/wordpress/?page_id=1378.

[7] (2014) Spletna dokumentacija brezžičnega modula ALPHA-TRX433S. Dostopno na http://www.rfsolutions.co.uk/acatalog/DS-ALPHA-TRX-7.pdf.

[8] (2014) Spletna dokumentacija družine mikrokrmilnikov STM32F407. Dostopno na http://www.st.com/web/en/resource/technical/document/reference_manual/DM00031020.pdf.

[9] (2014) Spletna dokumentacija razvojnega sistema STM32F4-Discovery. Dostopno na http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00039084.pdf.

[10] (2014) Spletna dokumentacija senzorja GP2Y0. Dostopno na http://www.erasme.org/IMG/gp2y0a02_e.pdf.